

An Adaptive Embedded Boundary Method for the Compressible Navier Stokes Equations

D. T. Graves

Applied Numerical Algorithms Group
Lawrence Berkeley National Laboratory

Berkeley, CA

April 7, 2015



Acknowledgements

- Lots of people have worked on Chombo EB: P. Colella, D. Devendran, A. Dubey, H. Johansen, T. Ligocki, P. McCorquodale, D. Martin, D. Modiano, J. Pilliod, P. Schwartz, D. Trebotich, B. Van Straalen, M. Barad, D. Calhoun, X. Gao, J. Johnson, B. Keen, G. Miller, A. Nonaka, J. Percelay, C. Shen, B. Sjogreen, and many others.
- Research supported financially by the Office of Advanced Scientific Computing, US Department of Energy, under contract number DE-AC02-05CH11231.



Overview

- Algorithm
 - Embedded boundary context.
 - Grid generation.
 - Strategies for stable discretizations of different classes of PDEs.
 - Layering these strategies for a Navier-Stokes algorithm.
 - Results.
- Chombo Software Infrastructure
 - AMR context
 - Chombo Tools for EB and AMR
 - Performance Results
- Ongoing work



Equations and Requirements

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0$$

$$\frac{\partial(\rho \mathbf{u})}{\partial t} + \nabla \cdot (\rho \mathbf{u} \mathbf{u} + p \mathbf{I}) = \nabla \cdot \boldsymbol{\tau}$$

$$\frac{\partial(\rho E)}{\partial t} + \nabla \cdot (\rho \mathbf{u} E + \mathbf{u} p) = \nabla \cdot (\boldsymbol{\tau} \mathbf{u}) + \nabla \cdot (\xi(\nabla T))$$

$$\boldsymbol{\tau} = \mu(\nabla \mathbf{u} + \nabla \mathbf{u}^T) + \lambda(\nabla \cdot \mathbf{u}) \mathbf{I}.$$

- The system of PDEs has both parabolic and hyperbolic terms.
- The domain contains complex geometry.
- We need strong conservation.
- We need an algorithm consistent with block-structured adaptivity.
- We want a time step that corresponds with the hyperbolic wave speeds.



Spatial Discretization

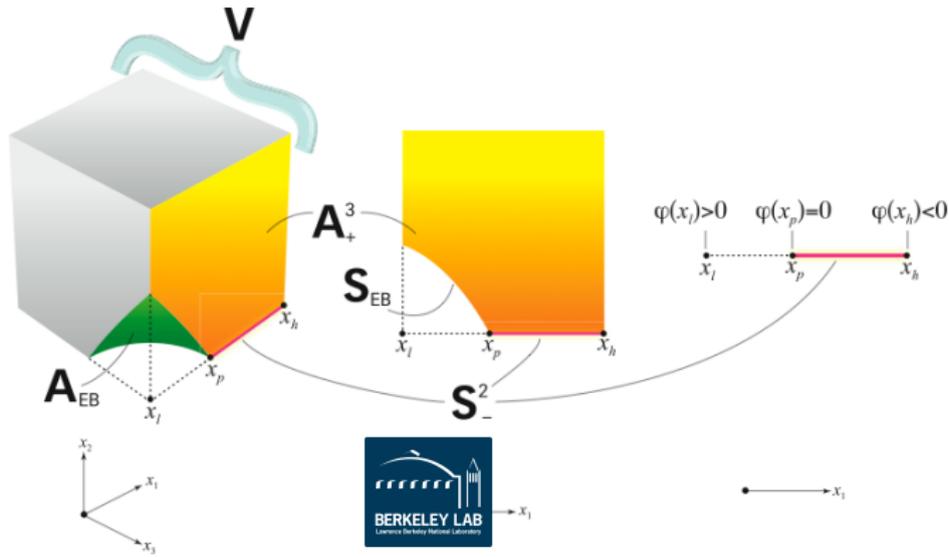


- Cartesian grid cells are cut with the surface of the geometry.
- Robust, fast, and accurate grid generation is tractable. Geometric information can be generated to any order of accuracy.
- Algorithms are complex but tools to manage this complexity are available.
- Load balancing requires some care.

EB Grid Generation

As input we take any implicit function $\phi(x, y, z) = 0$.

- The boundary surface is the zero set of the function.
- We move up the dimensions using the divergence theorem at each.
- For smooth ϕ , we can get grids of any accuracy (Schwartz, 2015).
- EB grid generation technology has been robust for a long time (Aftosis, Berger, Melton 1998).



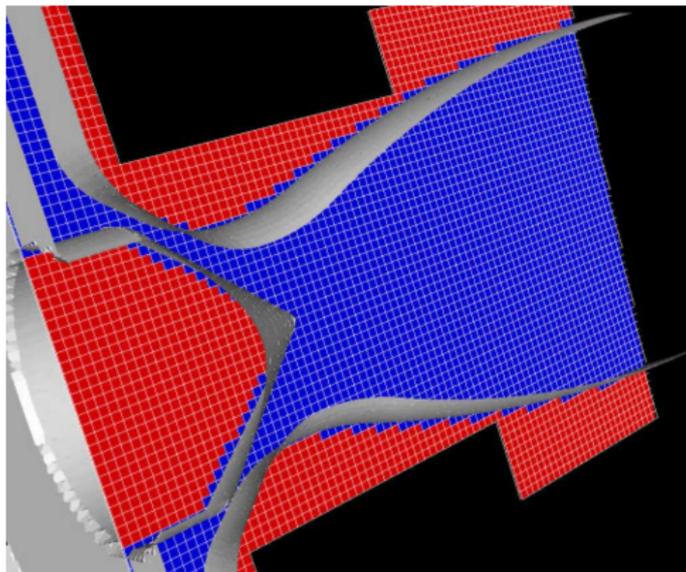
EB Grid examples

Grids can be generated using image data. This is calcite.



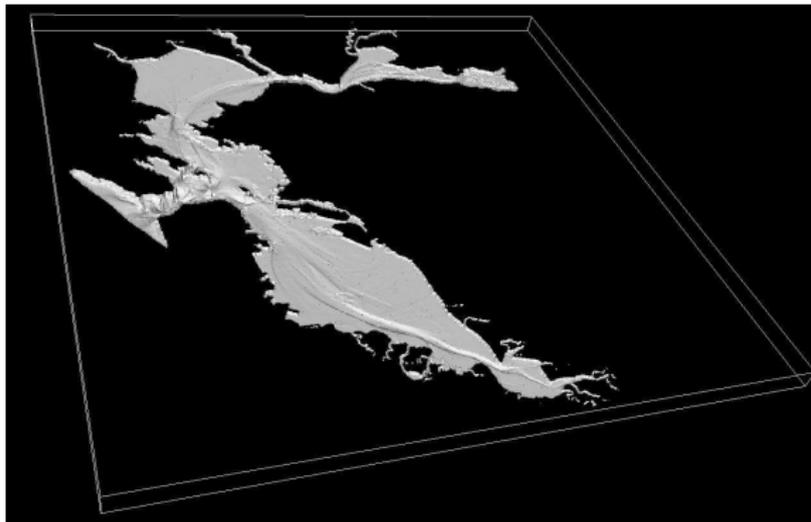
EB Grid examples

Grids can be generated using analytic functions. This is a nozzle from a wakefield accelerator.



EB Grid examples

Grids can be generated using local function values. This is the SF bay and delta.



Navier Stokes Dissection

- Divide the system into hyperbolic and parabolic terms.
- Most of the nonlinear terms are mercifully hyperbolic.
- Energy dissipation is the exception.

$$\frac{\partial U}{\partial t} + L^H(U) = L^E(U) \quad U = \begin{pmatrix} \rho \\ \rho \mathbf{u} \\ \rho E \end{pmatrix}$$

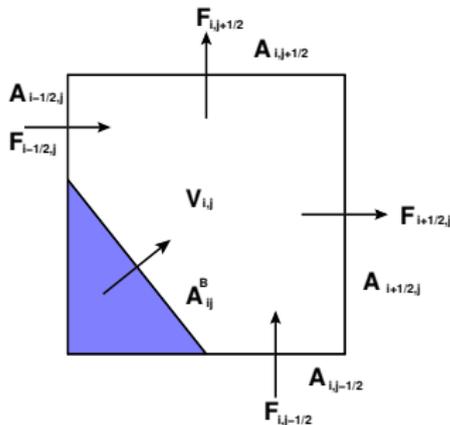
$$L^H = \begin{pmatrix} \nabla \cdot (\rho \mathbf{u}) \\ \nabla \cdot (\rho \mathbf{u} \mathbf{u} + pI) \\ \nabla \cdot (\rho \mathbf{u} E + \mathbf{u} p) \end{pmatrix} \quad L^E(U) = \begin{pmatrix} 0 \\ \nabla \cdot \tau \\ \nabla \cdot (\xi \nabla T) + \nabla \cdot (\tau \mathbf{u}) \end{pmatrix}$$

Discrete Divergence and Small Volumes

The divergence theorem, $\int_{V_i} \nabla \cdot F \, dV = \int_{\partial V_i} F \cdot \hat{n} \, dA$, becomes

$$D^c(F)_i = \frac{1}{\kappa_i h} \left(\sum_{d=1}^D (\alpha F^d)_i \mathbf{e}_d - (\alpha F^d)_{i-\frac{1}{2}} \mathbf{e}_d + (\alpha F)_i^B \cdot \hat{n}_i \right).$$

$$\kappa_i = |V_i|/h^D \quad \alpha_{i+\frac{1}{2}} \mathbf{e}_d = |A_{i+\frac{1}{2}} \mathbf{e}_d|/h^{D-1}, \quad \alpha_i^B = |A_i^B|/h^{D-1}$$



- The volume fraction κ can be arbitrarily small.
- We deal with this “small cell problem” differently for different classes of PDE.

Hyperbolic Equations with Small Cells

If you discretize a hyperbolic system of the form $\frac{\partial U}{\partial t} + \nabla \cdot F = 0$ as

$$U^{n+1} = U^n - \Delta t D^c F,$$

this has a time step constraint

$$\Delta t < \frac{h}{v^{max}} (\kappa_{min})^{\frac{1}{D}}$$

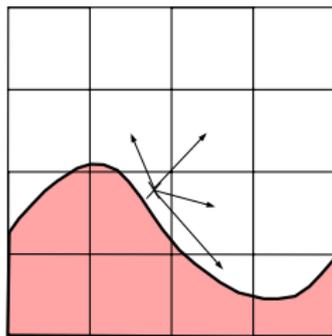
where v^{max} is the maximum wave speed and κ_{min} is the smallest volume fraction in the domain.

- Routinely, $\kappa_{min} = O(10^{-7})$ and is sometimes much smaller.
- Cell-merging methods eliminate small cells in grid generation.
- H-Box methods define larger control volumes.
- We use the oldest and simplest technique, redistribution.



Redistribution

- Get $D^{nc}(F)$, a stable, non-conservative approximation to $\nabla \cdot F$.
 - Update with hybrid divergence.
 - Redistribute δM to neighboring volumes.
 - Mass weighting works best with strong shocks.
- $U_{\mathbf{i}}^{n+1,*} = U_{\mathbf{i}}^n - \Delta t(\kappa_{\mathbf{i}} D^c(F) + (1 - \kappa_{\mathbf{i}})(D^{nc}(F)))$
 - $\delta M_{\mathbf{i}} = \kappa_{\mathbf{i}}(1 - \kappa_{\mathbf{i}})(D^c(F) - D^{nc}(F))$
 - $U^{n+1} = R(\delta M, U^{n+1,*})$



Non-conservative Divergence

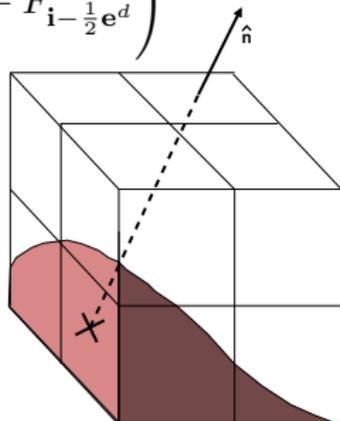
Here are two examples of common D^{nc} discretizations.

- Use the average of the neighboring D^c

$$D_i^{nc} = \frac{\sum_{j \in \mathcal{N}(i)} D^c(F)_j \kappa_j}{\sum_{j \in \mathcal{N}(i)} \kappa_j}$$

- Extrapolate to covered faces and use finite differences:

$$D^{nc} = \frac{1}{h} \left(\sum_{d=1}^{\mathcal{D}} F_{i+\frac{1}{2}e^d}^d - F_{i-\frac{1}{2}e^d}^d \right)$$



Elliptic Equations with Small Cells

For Poisson's equation

$$\nabla \cdot (\nabla \phi) = R,$$

to avoid small cell problems, we simply solve

$$\kappa D^c(\nabla \phi) = \kappa R.$$

- This diagonal weighting works quite well, even with geometric multigrid.
- Without this weighting, floating point problems can arise and multigrid smoothing becomes very difficult.



Parabolic Equations with Small Cells

Given the heat equation $\frac{\partial \phi}{\partial t} = \nu \nabla \cdot (\nabla \phi)$, with very small ν , one might be tempted to solve this explicitly.

$$\phi^{n+1} = \phi^n + \nu \Delta t D^c(\nabla \phi)$$

This is a bad idea, as the time step constraint for this is

$$\Delta t < \frac{h^2 (\kappa_{min})^{\frac{2}{D}}}{2D\nu},$$



Implicit Parabolic Equations

Implicit parabolic solvers with small cells tend to converge quickly because they are diagonally dominant. For the heat equation, $(L(\phi) \equiv D^c(\nabla\phi))$.

- Euler (first order): $(\kappa I - \Delta t \kappa L)\phi^{n+1} = \phi^n$
- Crank Nicolson (second order with occasional issues):
 $(\kappa I - \nu \Delta t \frac{1}{2} \kappa L)\phi^{n+1} = \kappa \frac{\nu \Delta t}{2} L\phi^n$.
- TGA (second order, very stable; $\{\mu_i\}$ is a set of constants):
 $\phi^{n+1} = (\kappa \nu I - \mu_1 \kappa L)^{-1}(\kappa)(\kappa I - \mu_2 \kappa L)^{-1}(\kappa I + \mu_3 \kappa L)\phi^n$.
- To preserve conservation, use the implicit advance to produce a stable approximation to $L(\phi)$:

$$L(\phi)^{n+\frac{1}{2}} = \frac{\phi^{n+1} - \phi^n}{\Delta t}.$$



Navier Stokes Outline

- Update the solution *explicitly* with hyperbolic terms.
- Compute the momentum diffusion *implicitly*.
- Compute the energy dissipation using both implicit and explicit techniques.
- Compute the conduction term *implicitly*.



Mass

- Compute U^* , the solution advanced explicitly using only hyperbolic terms. This includes the hybrid divergence and explicit redistribution.

$$U_{\mathbf{i}}^* = U_{\mathbf{i}}^n - \Delta t L_{\mathbf{i}}^H(U^n)$$

- This produces the final value of density ($\rho^{n+1} = \rho^*$).



Momentum

- Fixing density $\rho = \rho^{n+1}$, advance implicitly

$$\rho \frac{\partial \mathbf{u}}{\partial t} = \nabla \cdot \tau$$

one step from \mathbf{u}^* to produce a stable $L^m(\mathbf{u}) = \nabla \cdot \tau$.

$$(L^m(\mathbf{u}))^{n+\frac{1}{2}} = \rho \left(\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} \right),$$

- This gives the final value of momentum

$$(\rho \mathbf{u})^{n+1} = (\rho \mathbf{u})^* + \Delta t (L^m(\mathbf{u}))^{n+\frac{1}{2}}.$$



Energy Dissipation

We need to produce an approximation to $L^d = \nabla \cdot (\tau \mathbf{u})$. This is a nonlinear operator in \mathbf{u} , so a simple implicit solve is not possible.

- $F^s \equiv (\tau \mathbf{u})$.
- Compute $\kappa D^c(F^s)$ and $D^{nc}(F^s)$.
- Hybrid Divergence: $D^h(F^s) = \kappa D^c(F^s) + (1 - \kappa) D^{nc}(F^s)$.
- Advance energy from $(\rho E)^*$ using the hybrid divergence.
 $(\rho E)^{**} = (\rho E)^* + \Delta t D^h(F^s)$.
- Save the energy difference $\delta E_i = \kappa_i(1 - \kappa_i)(D^c(F^s) - D^{nc}(F^s))$ as a RHS for the temperature equation. This preserves conservation.



Conduction

- Advance the conduction equation from $T = T^{**}$. Fix $\rho = \rho^{n+1}$.

$$\rho C_V \frac{\partial T}{\partial t} = \nabla \cdot \xi \nabla T + \delta E$$

- Get a stable approximation to $L^k(T)^{n+\frac{1}{2}} = \nabla \cdot (\xi \nabla T)^{n+\frac{1}{2}}$:

$$L^k(T)^{n+\frac{1}{2}} = \rho C_v \left(\frac{T^{n+1} - T^{**}}{\Delta t} \right) - \delta E.$$

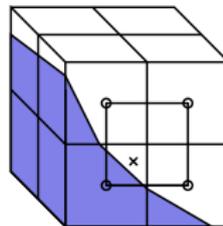
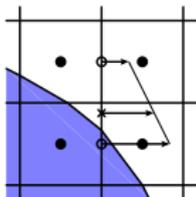
- This gives us the final value of energy

$$(\rho E)^{n+1} = (\rho E)^{**} + \Delta t (L^k(T))^{n+\frac{1}{2}}.$$



Flux Calculations

- Pointwise fluxes must be centered on face centroids for a second order accurate flux integral.
- We compute fluxes at face centers and interpolate to the centroids.



Different Operator's Fluxes

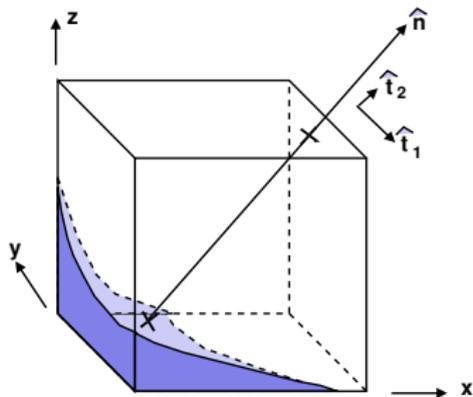
- For hyperbolic fluxes, we use a second order Godunov method. The pressure work is the only EB flux.
- For the conduction operator, ∇T is calculated at faces. and there is no EB flux (insulated boundary).
- For the viscous stress operator (and energy dissipation), the fluxes are constructed from $\nabla \mathbf{u}$. Since $\mathbf{u}_{EB} = 0$, $F_{EB}^s = 0$.
- The viscous flux at the EB is more complicated.



Viscous Momentum Flux

We need the $\nabla \mathbf{u}$ at the embedded boundary.

- $\frac{\partial \mathbf{u}}{\partial \hat{t}_1} = \frac{\partial \mathbf{u}}{\partial \hat{t}_2} = 0$ (no slip condition).
- We calculate the normal gradient with the $\mathbf{u} \cdot \hat{n} = 0$ and using surrounding values of \mathbf{u} .
- We rotate the resulting gradient to (x, y, z) .



Results–Convergence Test

We run a vortex with maximum Mach number $M = 0.5$ inside a sphere (or a circle in 2D) and measure solution error. All resolutions are run to a fixed time and compared to an exact answer (in this case, an even finer run). CFL is kept fixed.

2D convergence results:

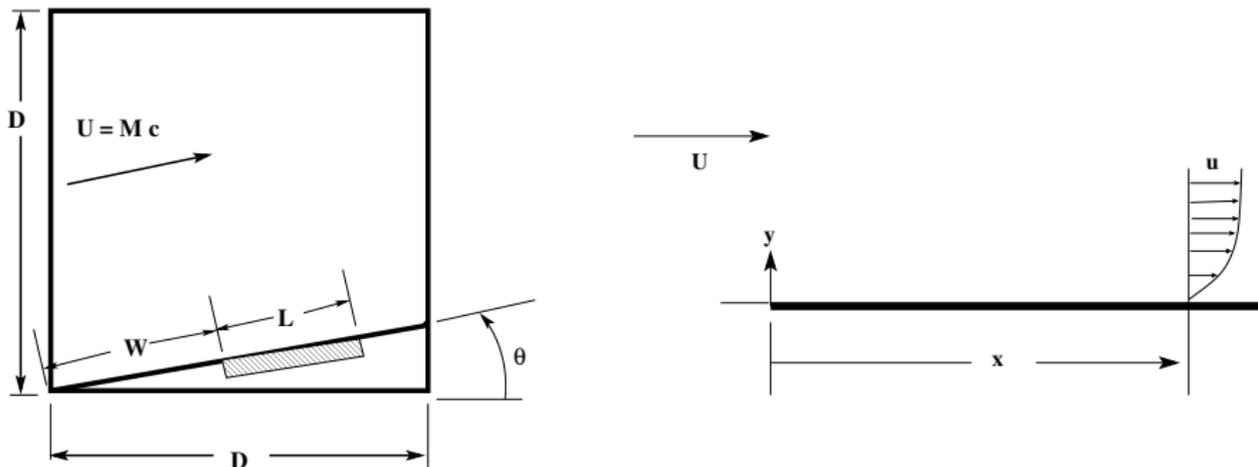
Variable	L_∞	L_1	L_2
(ρ)	1.980e+00	1.982e+00	1.978e+00
$(\rho\mathbf{u})_x$	1.822e+00	1.822e+00	1.824e+00
$(\rho\mathbf{u})_y$	1.822e+00	1.822e+00	1.824e+00
(ρE)	1.978e+00	1.978e+00	1.973e+00

3D convergence results:

Variable	L_∞	L_1	L_2
(ρ)	1.979e+00	1.986e+00	1.982e+00
$(\rho\mathbf{u})_x$	1.814e+00	1.805e+00	1.810e+00
$(\rho\mathbf{u})_y$	1.814e+00	1.805e+00	1.810e+00
$(\rho\mathbf{u})_z$	1.814e+00	1.805e+00	1.810e+00
(ρE)	1.978e+00	1.983e+00	1.980e+00

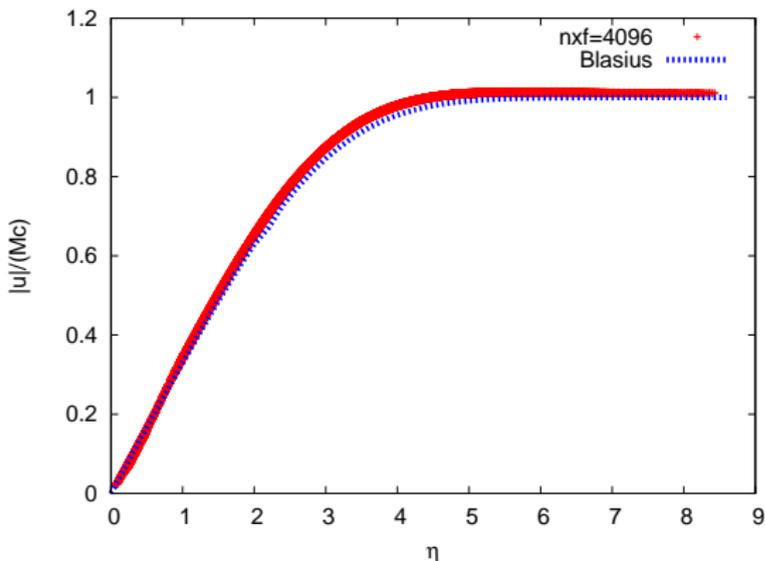
Results–Boundary Layer

We initialize a flow at constant velocity ($M = 0.5, Re_L = 30,000$) and measure the converged velocity as a function from the distance from where the no-slip condition starts.



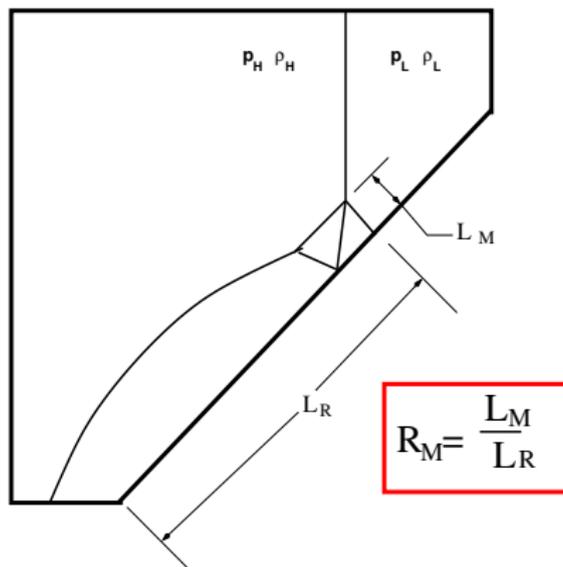
Boundary Layer vs. Blasius

We plot every point in the domain where $5000 < Re_x < 15000$ and show good agreement with the Blasius profile. We suspect that the departure might be due to compressibility effects. $\eta = y\sqrt{\frac{U}{\nu x}}$.



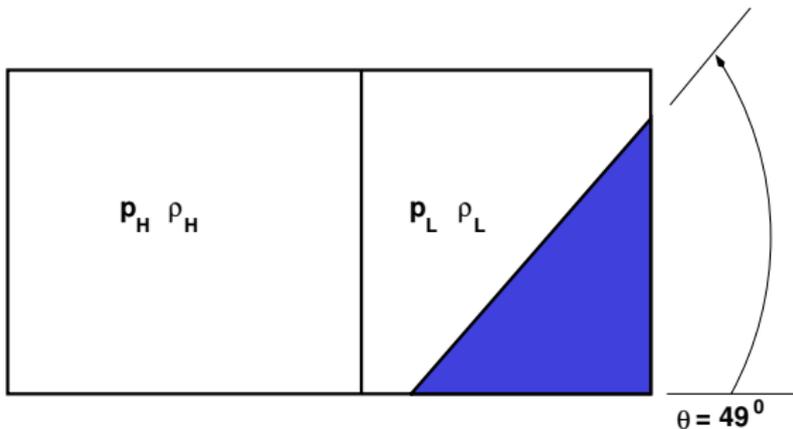
Glaz, et al. Shock Reflection

Glaz, et al. (1985) compared an inviscid, regular grid calculation to an experimental shock reflection at $M = 7.1$. They found the results were very different. The ratio R_M was much smaller in the experiment.



Results– Shock Reflection

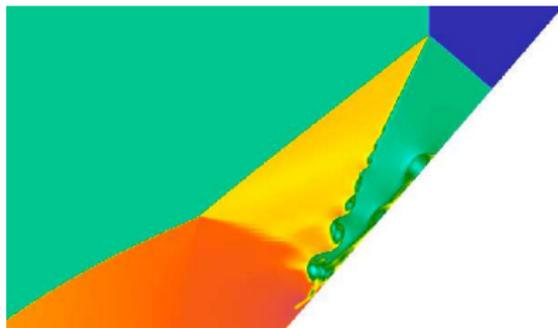
We make a shock tube to produce a $M = 7.1$ shock over a wedge to compare with Glaz, et al. (128×64 base grid, 7 levels of refinement, 16384×8192 effective resolution at the finest level).



Results– Mach stem length, inviscid

Glaz, et al.: $R_M = 0.07$ (grid aligned calculation)

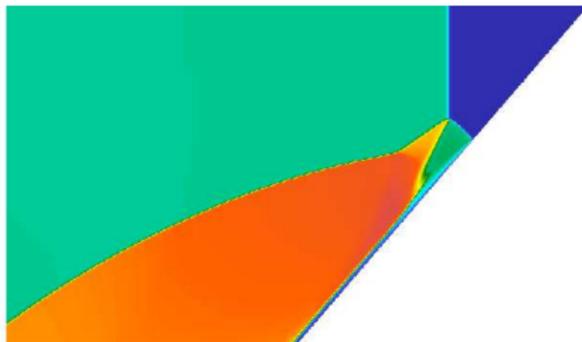
Us: $R_M = 0.072$ (EB calculation).



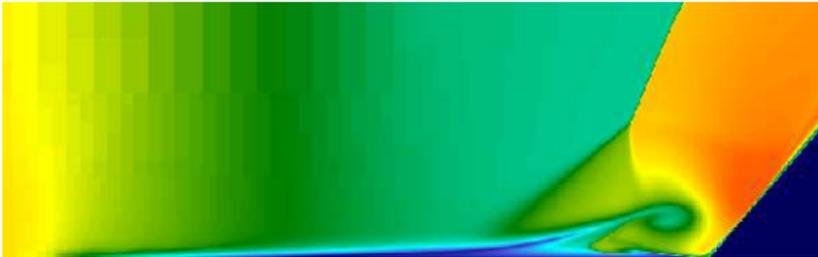
Results– Mach stem length, viscous

Glaz et al.: $R_M = 0.038$ (experiment)

Us: $R_M = 0.03$ (calculation)



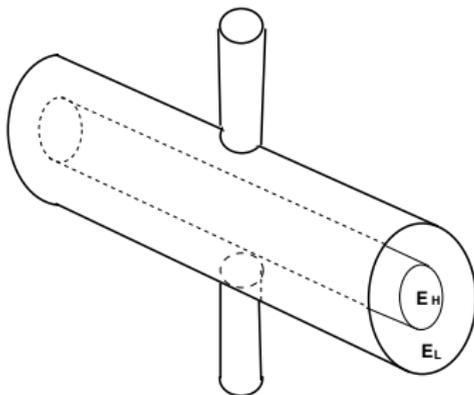
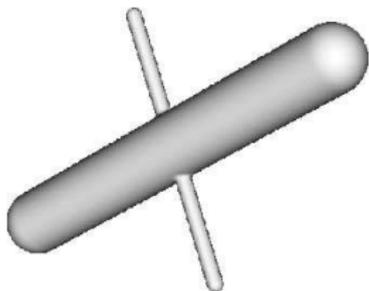
Results– Lambda Shock



We also captured a classic shock-boundary layer interaction pattern, the lambda shock. See Schlichting for details.

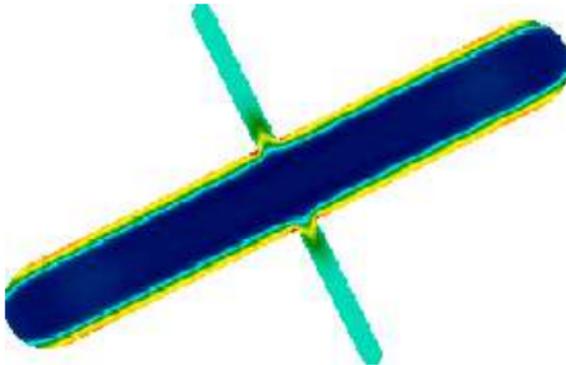
Wakefield Capillary Tube

In the capillary tubes of wakefield accelerators they energize the core. The core expands to make a “fluid pipe”. This simulation is meant to determine whether the fill tubes distort this pipe. We do not include ionization or magnetization. Initially $\rho = \rho_0$, $\mathbf{u} = 0$.



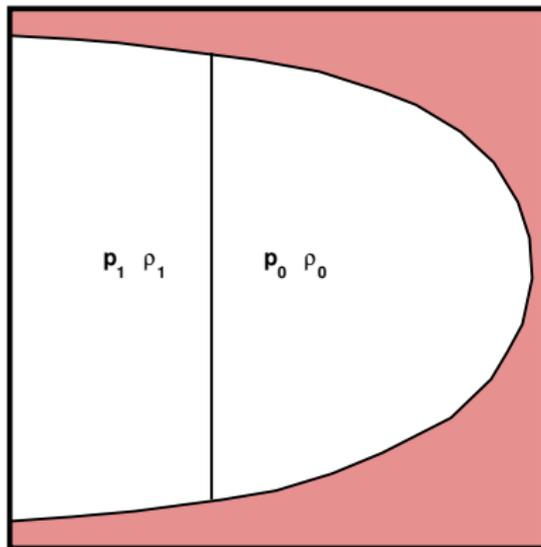
Wakefield Capillary Tube

We do observe a formation of a low density core and it does, unsurprisingly, distort rather quickly ($35\mu\text{s}$) around the fill tubes.



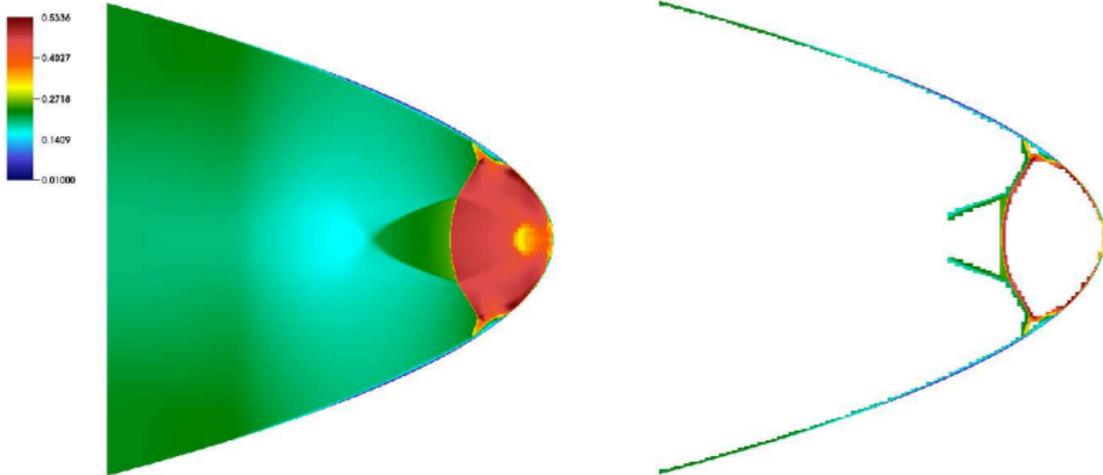
Parabolic Mirror

To illustrate AMR efficiency gains, we set up a shock tube in front of a parabolic mirror. 4 level calculation (coarsest 128^D , finest 1024^D)



Parabolic mirror

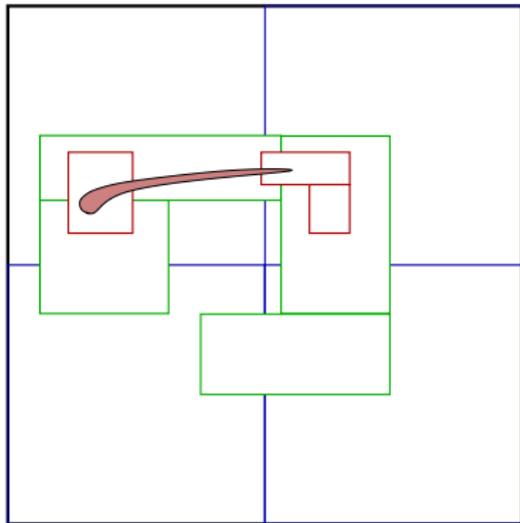
The finest level only occupies a small percentage of the domain even after many shock reflections.



Chombo EB AMR

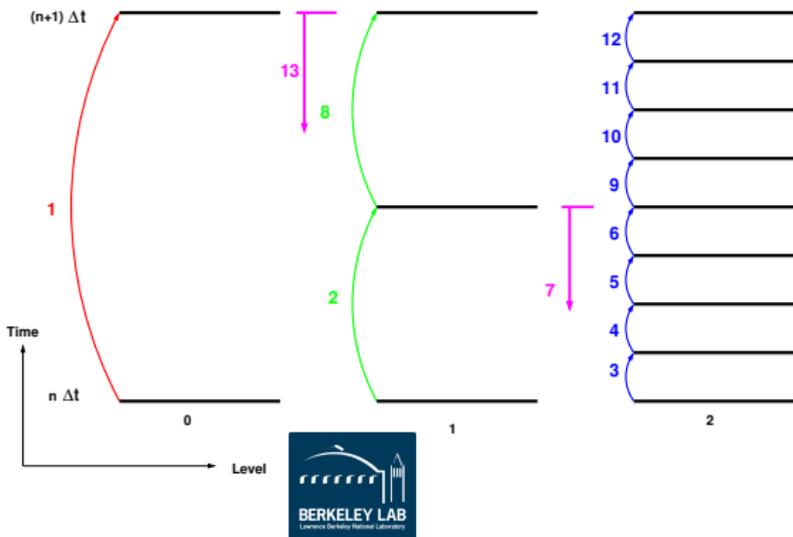
Chombo provides infrastructure for block-structured AMR and supports particles, mapped grids, multiblock....

- AMR levels are represented as unions of boxes. Each box gets a processor assignment.
- Chombo has very usable frameworks for elliptic and time-dependent AMR.
- EB presents some unique challenges. So let's focus on EB.



Chombo Time Stepping

- Berger-Oliger (1984) recursive time stepping allows all refinements to have the same CFL.
- Coarse levels are advanced first and used as boundary conditions for finer advances.
- Level synchronizations (refluxing, averaging, elliptic constraints) happen when levels reach the same time.



Chombo Elliptic Framework

- AMRMultigrid follows the Martin-Cartwright (1996) algorithm for geometric multigrid in an AMR context.
- Templated framework allows any linear operator.
- Other linear solvers (eg. BiCGStab) are included.
- An interface to PETSc is included.
- Geometric multigrid preconditioned PETSc (which uses algebraic multigrid) is a common choice.

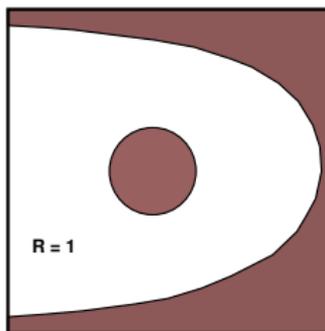


Chombo's EB Elliptic Operators

The EB operators included with Chombo are

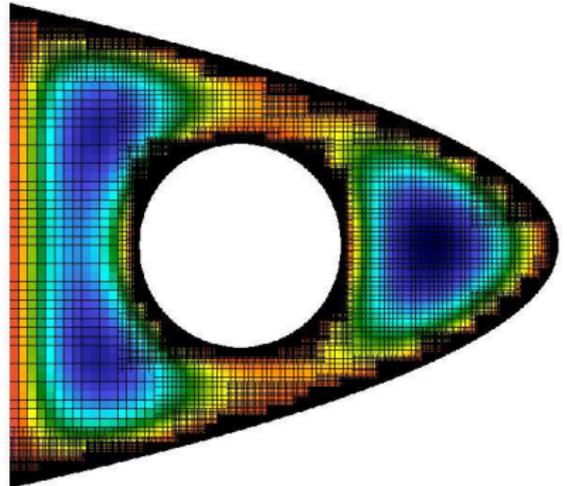
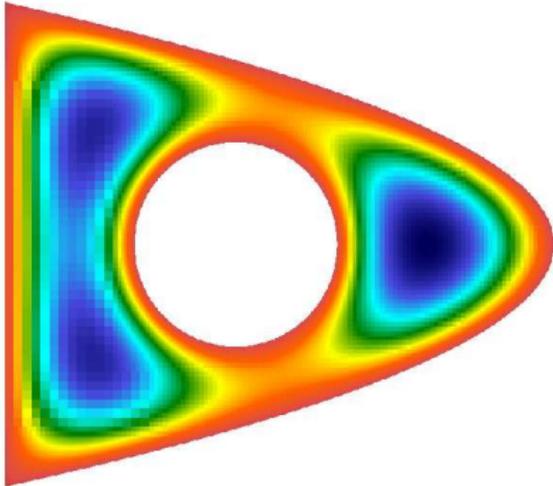
- constant coefficient Poisson,
- variable coefficient Helmholtz,
- variable coefficient viscous tensor.

To demonstrate, we solve $\nabla \cdot \nabla \phi = 1$, $\phi = 0$ on all boundaries.



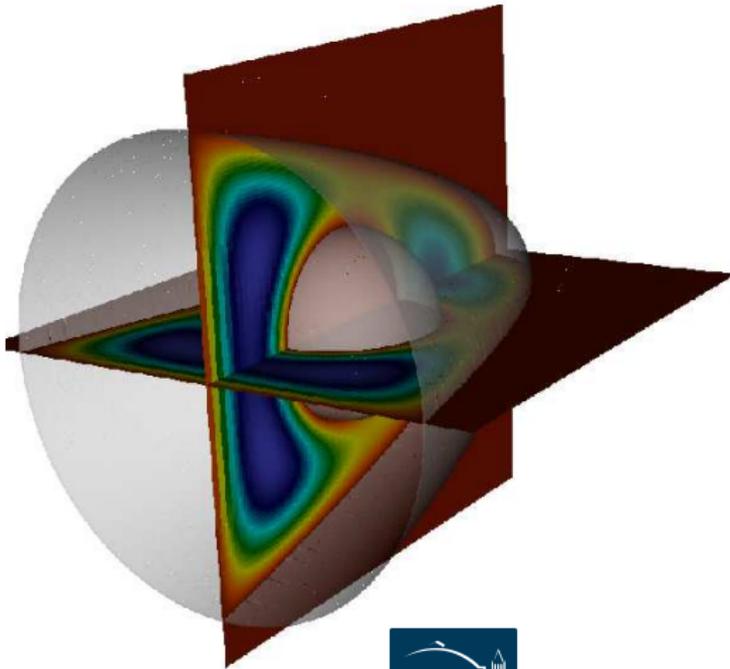
Poisson in 2D

64^2 , 4 levels of refinement, effective resolution 1024^2 . Run on a desktop in less than a minute.



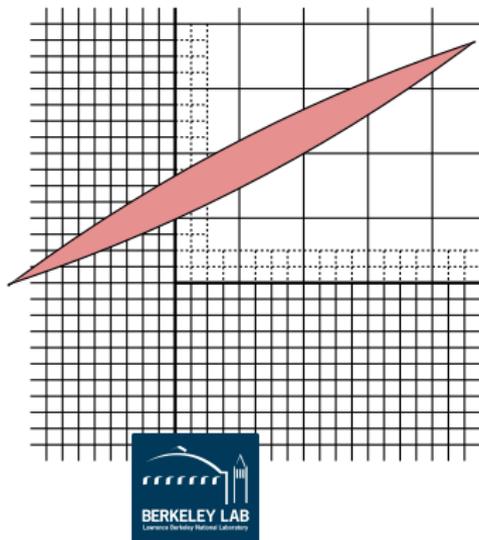
Poisson in 3D

64^3 , 2 levels of refinement, effective resolution 256^3 . Run on a desktop in a couple of minutes.



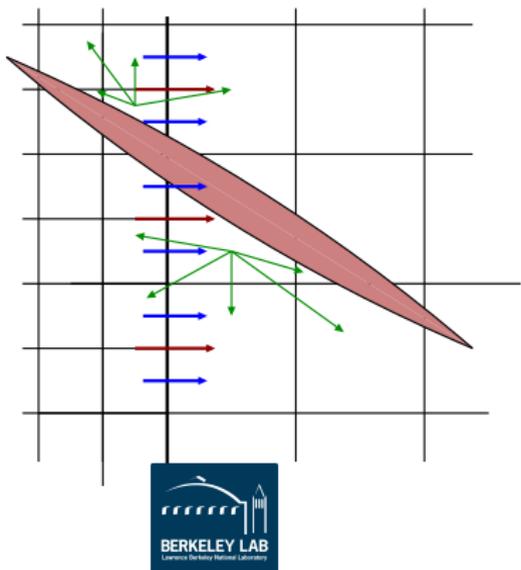
Coarse-Fine Interpolation

- We fill ghost cells for elliptic problems using an extension of (Schwartz, et al. 2006).
- This algorithm can get a bit arcane considering we have to avoid using coarse data covered by finer data.
- Coarse-fine interpolation gets much easier with the higher-order EB formulation.



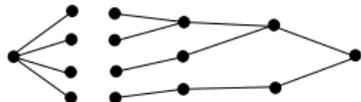
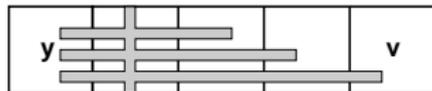
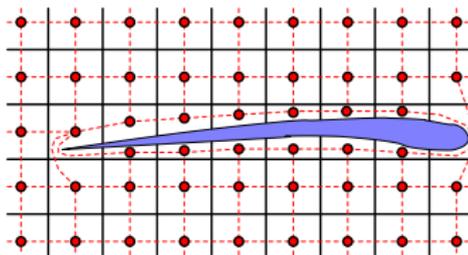
Redistribution and Refluxing

- Refluxing preserves conservation through flux matching.
- Redistribution weights can be based on volume alone or have a weighting. For CNS, we use mass-weighting.
- The bookkeeping for redistribution has to take into account mass that leaves the level. Chombo has tools for this, as well.



Chombo EB Description

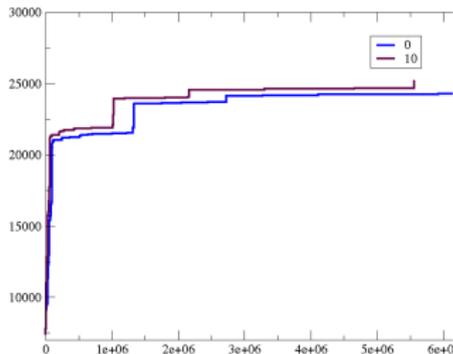
- Connectivity at every resolution is described by a sparse, distributed graph.
- The graph is formed at the finest level and coarsened using graph coarsening.
- This graph allows for multiple volumes/cell and can be complex.
- Heterogeneous description presents performance challenges.



EB Performance Measurement

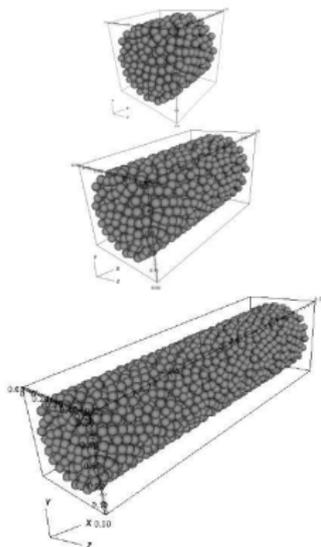
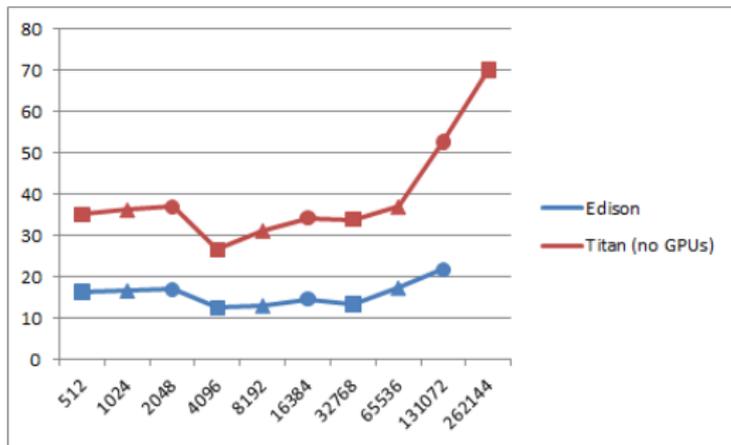
- Chombo timers present a good view of the time spent in different routines with very low overhead.
- Chombo memory counters can provide a very accurate picture of how much memory has been allocated (and what the OS thinks is the memory usage). The trace can be used to locate spots in the code where the memory jumps.

EBAMRCNS::advance	166.26	10	
98.5%	163.83	10	diffusion dance
1.3%	2.23	10	flux divergence
0.1%	0.16	10	time stepcalc
0.0%	0.02	10	copyTo
0.0%	0.01	10	max min check



EB Parallel Performance

- We have gotten good weak scaling to $O(10^8)$ processors with EBAMRINS using a timed load balance.
- Memory imbalance can be an issue.



Ongoing EB Performance Work

- We are exploring using memory as the load for load balancing.
- Using execution time as a load has been effective.
- Different modules have to be load balanced differently.
- Introspective load balancing with hybrid MPI/OpenMP will probably not be sufficient forever. Other strategies being explored include:
 - Tiling (TIDA).
 - Runtime systems (Charm++, GASNet) to avoid synchronizations.
 - Stencil-based languages.
 - Higher order algorithms to increase arithmetic intensity.



Higher Order EB Algorithm Development

Define a flux F as a polynomial expansion around a coordinate face f .
The flux integral becomes:

$$\int_f F \, dA = \sum_{|p| < P} c_p \int_f (\mathbf{x} - \bar{\mathbf{x}})^p \, dA$$

- We define area, volume moments to be $m_A^p \equiv \int_f (\mathbf{x} - \bar{\mathbf{x}})^p \, dA$, $m_V^p \equiv \int_V (\mathbf{x} - \bar{\mathbf{x}})^p \, dV$.
- These moments emerge directly from the grid generation algorithm and can be calculated once and cached.
- Higher order algorithms reduce to finding the polynomial coefficients $\{c_p\}$.
- At the cut face (where $\hat{n} \neq \text{const}$), this gets more complicated, but manageably so.



Stencils for Higher Order EB

Given cell-averaged data $\phi_{\mathbf{i}} = \frac{1}{V_{\mathbf{i}}} \int_{V_{\mathbf{i}}} \phi \, dV$, and a neighbor set \mathcal{N} around a face f we get a system of equations for $\{c_p\}_f$. For $\mathbf{j} \in \mathcal{N}$,

$$\sum_{|p| < P} c_p m_{\mathbf{j}}^p = \phi_{\mathbf{j}}$$

- We solve the over-determined system using weighted least squares.
- Careful choice of weighting function is key to getting a stable algorithm.
- Fourth-order Poisson is complete (Devendran, submitted 2014). We are working on incompressible Navier Stokes.
- Coarse-fine interpolation is much simpler with this formulation since averaging is exact.



Conclusions

- Embedded boundary methods have fairly simple grid generation but the algorithms can be complex.
- Hyperbolic, elliptic and parabolic equations all have different associated strategies for solving small cell problems.
- These strategies can be layered for solving more complex systems.
- Chombo provides tools for managing AMR and EB complexity.
- All this software, including the compressible Navier Stokes code, is available at <http://chombo.lbl.gov>. This is free software under a BSD license.
- EB algorithm and performance research continues apace.
- Thank you.

